

## Commands

<Ctrl-C>	stop running program
auto [line]	automatically number program lines
clear [flash]	clear ram [and flash] variables
cls	clear terminal screen
cont [line]	continue program from stop
delete ([line][-line]) subname)	delete program lines
dir	list saved programs
edit line	edit program line
help [topic]	online help
list ([line][-line]) subname)	list program lines
load name	load saved program
memory	print memory usage
new	erase code ram and flash memories
profile ([line][-line]) subname)	display profile info
purge name	purge saved program
renumber [line]	renumber program lines (and save)
reset	reset the MCU!
run [line]	run program
save [name]	save code ram to flash memory
undo	undo code changes since last save
upgrade	upgrade StickOS firmware!
uptime	print time since last reset

## Modes

analog [millivolts]	set analog voltage scale
autorun [on off]	autorun mode (on reset)
baud [rate]	* UART transport baud rate (on reset)
echo [on off]	terminal echo mode
indent [on off]	listing indent mode
ipaddress [dhcp ipaddress]	set/display ip address
nodeid [nodeid none]	set/display zigflea nodeid
numbers [on off]	listing line numbers mode
pins [assign [pinname none]]	set/display pin assignments
prompt [on off]	terminal prompt mode
servo [Hz]	set/display servo Hz (on reset)
sleep [on off]	debugger sleep mode
step [on off]	debugger single-step mode
trace [on off]	debugger trace mode
usbhost [on off]	set/display USB host mode (on reset)
watchsmart [on off]	low-overhead watchpoint mode

## General Statements

line	delete program line
line statement	enter program line
assert expression	break if expression is false
data n [, ...]	read-only data
dim variable[\$][[n]] [as ...], ...	dimension variables
end	end program
halt	* loop forever
input [dec hex raw] variable[\$], ...	* input data
label label	read/data label
let variable[\$] = expression, ...	assign variable
print [dec hex raw] expression, ...	print strings/expressions
read variable [, ...]	read data into variables
rem remark	remark
restore [label]	restore data pointer
sleep expression (s ms us)	delay program execution
stop	insert breakpoint in code
vprint var[\$] = [dec hex raw] expr, ...	* print to variable

## Pins

Use the "help pins" command to see MCU-specific pin names and capabilities; use the "pins" command to set/display pin assignments

# StickOS Quick Reference (v1.80\*)

<http://www.cpushick.com>

## Block Statements

```
if expression then
[elseif expression then]
[else]
endif
```

```
for variable = expression to expression [step expression]
[(break|continue) [n]]
next
```

```
while expression do
[(break|continue) [n]]
endwhile
```

```
do
[(break|continue) [n]]
until expression
```

```
gosub subname [expression, ...]
```

```
sub subname [param, ...]
[return]
endsub
```

## Device Statements

timers:

```
configure timer n for n (s|ms|us)
on timer n do statement
off timer n                disable timer interrupt
mask timer n              mask/hold timer interrupt
unmask timer n            unmask timer interrupt
```

uarts:

```
configure uart n for n baud n data \
(even|odd|no) parity [loopback]
on uart n (input|output) do statement
off uart n (input|output)  disable uart interrupt
mask uart n (input|output) mask/hold uart interrupt
unmask uart n (input|output) unmask uart interrupt
uart n (read|write) variable, ... * perform uart I/O
```

\* i2c:

```
i2c start addr                master i2c I/O
i2c (read|write) variable, ...
i2c stop
```

qsqi:

```
qsqi variable [, ...]        master qsqi I/O
```

watchpoints:

```
on expression do statement
off expression                disable expr watchpoint
mask expression              mask/hold expr watchpoint
unmask expression            unmask expr watchpoint
```

## ZigFlea

```
<Ctrl-D>                    disconnect from remote node
connect nodeid                connect to remote node via zigflea
```

remote node variables:

```
dim varremote[[n]] as remote on nodeid nodeid
```

## Expressions

the following operators are supported as in C, in order of decreasing precedence:

n	decimal constant
0xn	hexadecimal constant
'c'	* character constant
variable	simple variable
variable[expression]	array variable element
variable#	* length of array or string
( )	grouping
! ~	logical not, bitwise not
* / %	times, divide, mod
+ -	plus, minus
>> <<	shift right, left
<= < >= >	inequalities
== !=	equal, not equal
^ &	bitwise or, xor, and
^^ &&	logical or, xor, and

## \* Strings

v\$ is a nul-terminated view into a byte array v[]

string statements:

```
dim, input, let, print, vprint
if expression relation expression then
while expression relation expression do
until expression relation expression
```

string expressions:

```
"literal"                literal string
variable$                variable string
variable$(start:length) variable substring
+                        concatenates strings
```

string relations:

```
<= < >= >                inequalities
== !=                    equal, not equal
~ !~                      contains, does not contain
```

## Variables

all variables must be dimensioned!  
variables dimensioned in a sub are local to that sub  
simple variables are passed to sub params by reference  
array variable indices start at 0  
v is the same as v[0], except for input/print/i2c/qsqi statements

ram variables:

```
dim var[$][[n]]
dim var[[n]] as (byte|short)
```

flash parameter variables:

```
dim varflash[[n]] as flash
```

pin alias variables:

```
dim varpin[[n]] as pin pinname for \
(digital|analog|servo|frequency|uart) \
(input|output) \
[debounced] [inverted] [open_drain]
```

absolute variables:

```
dim varabs[[n]] at address addr
dim varabs[[n]] as (byte|short) at address addr
```

system variables (read-only):

```
* getchar    nodeid
msecs       seconds
ticks       ticks_per_msec
```